DISCOVERING SYSTEM HEALTH ANOMALIES USING DATA MINING TECHNIQUES

Ashok N. Srivastava, Ph.D.
Discovery and Systems Health Technical Area
Intelligent Systems Division
NASA Ames Research Center
Mail Stop 269-4, Moffett Field, CA 94035
ashok@email.arc.nasa.gov

ABSTRACT

We present a data mining framework for the analysis and discovery of anomalies in high-dimensional time series of sensor measurements that would be found in an Integrated System Health Monitoring system. We specifically treat the problem of discovering anomalous features in the time series that may be indicative of a system anomaly, or in the case of a manned system, an anomaly due to the human. Identification of these anomalies is crucial to building stable, reusable, and cost-efficient systems. The framework consists of an analysis platform and new algorithms that can scale to thousands of sensor streams to discovers temporal anomalies. We discuss the mathematical framework that underlies the system and also describe in detail how this framework is general enough to encompass both discrete and continuous sensor measurements. We also describe a new set of data mining algorithms based on kernel methods and hidden Markov models that allow for the rapid assimilation, analysis, and discovery of system anomalies. We then describe the performance of the system on a real-world problem in the aircraft domain where we analyze the cockpit data from aircraft as well as data from the aircraft propulsion, control, and guidance systems. These data are discrete and continuous sensor measurements and are dealt with seamlessly in order to discover anomalous flights. We conclude with recommendations that describe the tradeoffs in building an integrated scalable platform for robust anomaly detection in ISHM applications.

INTRODUCTION

Modern ISHM systems contain hundreds or thousands of sensors producing discrete and continuous measurements. The union of all the sensor signals at a given time t can be considered the observed state of the system. The entire record of the sensor measurements represents the evolution of the system through time. In this paper we focus on the situation where the sensor measurements are producing discrete signals. Discovering anomalies in continuous systems has been extensively treated in the data mining and statistics literature [5, 6]. We assume that the observed system evolution can be functionally described by the following equations:

$$\mathbf{h}_t = \Phi(\mathbf{h}^*_{t-1}) \quad (1)$$

$$\mathbf{x}_t = \Psi(\mathbf{x}^*_{t-1}, \mathbf{h}^*_{t-1}, u_t) \quad (2)$$

We assume that the function Φ determining the evolution of the system state $\mathbf{h}_t$ is unknown. We also discuss two situations, one where the state $\mathbf{h}_t$ is observable, and one where it is assumed to be hidden. The vector $\mathbf{x}$ is an N dimensional observed binary state vector, and $\mathbf{x}^*_{t-1}$ is the entire history of the observed state vector: $\mathbf{x}^*_{t-1} = [\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_{t-1}]$. The quantity u_t is the observed input to the system. Because we assume that we do not know the function Ψ, we cannot rely on it to help us determine whether or not the observed vector $\mathbf{x}_t$ is anomalous.

The problem that we address in this paper is to develop a method to discover whether or not the current observed state $\mathbf{x}_t$ is anomalous or not based on the observed history of the system, as well as replicates of the system behavior. Thus, let $\mathbf{X}$ be a $T \times N$ matrix whose tth row $X(t,:)$ contains the values of the N binary sensors at time t during a single run of the system. Likewise, the nth column $X(:,n)$ contains the time series representing the time-ordered sequence of states of the nth sensor. Different replicates of the system are indexed by l and are represented by different $\mathbf{X}_l$, for $l = 1 \ldots L$.

We make two assumptions regarding the sensor measurements in order to develop the ideas in this paper:

- the times in each $\mathbf{X}_l$ have all been normalized so that $t = 0$ (i.e. the first row of $\mathbf{X}_l$) corresponds to the start of the system.
- all replicates of the system l under consideration correspond to the the system evolving under the same basic operating conditions.

In order to ground this example to a real system, we consider the situation where there are $N \approx 1000$ different binary switches in an aircraft cockpit [1]. We assume that we have a recording of length $T \approx 1000$ of the switch settings, sampled at uniform intervals. Note, however, that each flight may have a different value of T, since different flights have different durations, etc. As the pilot maneuvers the airplane through its various phases of flight, the pilot flips these switches. We assume that we have $L \approx 10,000$ different flights recorded.

Clearly, the vast majority of the flights in both civilian and military aircraft are such that the pilot flips the appropriate switches at the appropriate times in the appropriate *order*. Due to standard operating procedures (SOPs) and extensive training, one would expect the deviation between different flights to be small.

This paper addresses the problem of identifying anomalies in the sequences of states of the switches that are observed. The anomaly is only defined with respect to the sample of L different flights, or replicates, since we do not have an explicit model of how the system evolves through time, i.e., we do not, and cannot know Φ or Ψ in Equations 1 and 2.

Note that the data sets involved in this simple example are extremely large. For the values given above, the data set $D = \{\mathbf{X}_1, \cdots, \mathbf{X}_L\}$ is on the order of 1.25 gigabytes in size. In the event that the data set D is refreshed daily, we have an incredible escalation in data set size and associated storage and maintenance costs. The methods that we develop need to be able to scale to the massive data sets without requiring supercomputing power and significant storage media. With these assumptions we would like to build a model $P(\mathbf{X})$ from a data set $D = \{\mathbf{X}_1, \cdots, \mathbf{X}_L\}$. The data $\mathbf{X}_l$ arise from different airlines flying the same route using the same airplane. Given that we are going to have a large volume of data any model for $P(\mathbf{X})$ must be able to learned and probed quickly. [2]

TWO MODELS OF THE DATA GENERATING PROCESS

We next describe two models of the data generating process that obey Equations 1 and 2, but that have very different statistical characteristics, and therefore very different implications on the ISHM problem. The first model assumes that the data are generated by a process where each observation is probabilistically independent of each other observation, and that the state $\mathbf{h}_t$ is fully observed. Since the state vector is fully observed, we can assume that it is part of the observation vector $\mathbf{x}_t$ without loss of generality. The independence can be expressed as the following expression, for all t, t', n, n':

$$P(\mathbf{x}_t(n), \mathbf{x}_{t'}(n')) = P(\mathbf{x}_t(n))P(\mathbf{x}_{t'}(n')) \tag{3}$$

This data set represents a simple 'baseline' model which we can use to evaluate the performance of the algorithms that we describe in this paper. In order to generate data that matches this assumption, we create a $T \times N$ data set where $(T = 1000, N = 500)$ using binomial random variables with parameters $(n = 1, p = 0.3)$. We will refer to this data set as the *baseline data set*. A key aspect of this data set is that we assume that the sensor measurements are a full representation of the state of the system. Thus, methods that are successful in finding faults in this data set would be those that do some sort of envelope detection either on the sensor measurements themselves or a transformed representation of the measurements. This data set will illustrate the properties of

[1] Note that switches with m multiple settings can always be converted to m binary signals.

[2] As the data will be accumulated daily it might also be important that the model can easily be updated incrementally.

envelope detection algorithms that are commonplace in the ISHM literature.

The second model of the data generating process is much more complex, but it allows for unobserved system anomalies and dynamically evolving states and observable vectors. We will refer to this data set as the *dynamic data set*. The model is based on the Hidden Markov Model, which is widely used in speech recognition [7] and has also been discussed in the IVHM/ISHM literature [1, 3]. We next briefly describe the Hidden Markov Model and its applications to IVHM.

The HMM is a dynamic model consisting of H hidden states that are assumed to be not directly observable, with S observation symbols associated with each state. The model has an initial probability distribution π which is an $S \times 1$ vector where π_i is the probability that the system begins in state i. The HMM starts in an initial state according to the distribution π and then the hidden state h_t moves to the next state based on a Markov transition matrix A. The matrix $A_{ij} = P(h_{t+1} = j | h_t = i)$ which gives the probability distribution of the next hidden state given the current state. While the system is in state i it is assumed to emit an observation vector $\mathbf{x}_t$ according to the distribution $B_{jk} = P(\mathbf{x}_t = k | h_t = j)$.

Notice that the HMM assumes a discrete state space for the hidden variable $\mathbf{h}_t$, and that it is assumed that the observation vector $\mathbf{x}_t$ is also discrete. In our case, the number of symbols $S \approx 2^N$ where $N = 500$. Thus, we need to perform a preprocessing step to identify a small number of representative states. This step is as much art as science, and is usually performed through the use of a clustering algorithm. Banerjee et. al. [2] have given an excellent formulation to the problem of clustering high dimensional data based. We use their algorithm along with standard clustering methods such as the k-means algorithm for grouping observations into symbols.

The implications of the HMM formulation for ISHM problems in interesting. The HMM allows for modeling situations where we assume that the system state is evolving according to an unobserved set of dynamics defined by the Markov transition matrix. As the system evolves, the state may go into one or more 'failure modes' that not directly observable at the sensors. However, based on the distribution of observed sensor measurements, it may be possible to determine a failed state. This is the avenue which we explore in this paper. In the simulations for this paper, we chose $H = 4$, $S = 6$, and the probability matrices A and B such that the system can fall into a failure mode (state 6) with probability 0.05. When the HMM construct is applied to real data, these values will generally be much larger.

As noted above, the data set size is potentially very large. For the systems described in this paper, the data is provided as the set of matrices $D = \{\mathbf{X}_1, \cdots, \mathbf{X}_L\}$. The first observation that can be made is that the matrices $\mathbf{X}_l$ are all highly sparse, meaning that most of the data is all not changing at any given time. Thus, we converted the data into a very compact representation as follows:

1. Record the initial values for all binary variables $\mathbf{x}_0$. This vector represents the initial state of the system at time 0.
2. Record only those variables that transition from the initial state to the next state. Since the variables are all binary, this can be recorded simply as the index number of the switch that changed.
3. If needed, record the time at which the transition occurred.

This representation of the data reduces the data set size significantly. Our experiments indicate that for real data sets from approximately $10,000$ flights, the data set size drops to approximately 1% of the original size without any loss of information. However, this data representation imposes certain constraints on the underlying algorithms. We will discuss this tradeoff in the next sections.

MODELING APPROACH

At opposite extremes there are two potential simplifications to make regarding this problem. We could focus either on the temporal aspects of the problem (non-stationary behavior across rows), or on the correlation aspects between the sensors (stationary behavior across columns). The most general kind of model, where we look at correlations (or higher order analogues) between $X(t, n)$ and $X(t', n')$, we discard as being to complex as a starting point. To obtain initial insight quickly we focus on a simple model for the time dependence and a simple model for the stationary behavior. However, our proposed approach can be extended to relax these

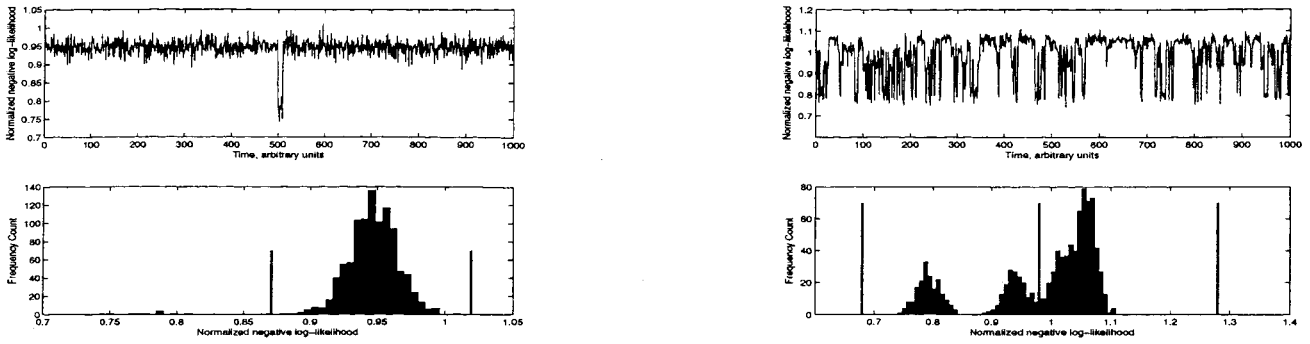

Figure 1: We computed the likelihood of each observation for both the benchmark data set and the dynamic data set as a function of time. A failure mode in the benchmark data set is apparent in the drop in the likelihood of the points around time step 500. These are shown in the top panel of plots for both data sets respectively. Notice that most of the data lies within the ± 3 sigma confidence interval of the mean likelihood as indicated in the lower panel. The unimodal characteristic of the benchmark data set is clear in the lower left hand plot. The multimodal characteristic of the observation sequence is also apparent in the lower right hand plot. The observations are not indicative of a failure state.

simplifying assumptions.

In the simplest approximation we completely ignore the time dependence, and build a model for the likelihood of any particular row $\mathbf{x}$ of $\mathbf{X}$. Different rows (i.e. different times) are assumed to be independent so that $P(\mathbf{X}) = \prod_t P(\mathbf{x}_t)$ where t ranges over the rows of $\mathbf{X}$ and $\mathbf{x}_t = \mathbf{X}(t,:)$. Thus, we simplify things to the point of only having to build a model for $P(\mathbf{x})$ (i.e. for a single row).

INDEPENDENT SENSORS INDEPENDENT OF TIME

The most naive model for an observation $\mathbf{x}$ assumes independence between columns (sensors), i.e.

$$P(\mathbf{x}) = \prod_{n=1}^{N} P(x(n)) \tag{4}$$

where $x(n)$ is the nth component of $\mathbf{x}$. For binary data a Bernoulli assumption can be used to model the value of the bit $x(n)$ so that

$$P(x(n)) = \rho_n^{1-x(n)}(1-\rho_n)^{x(n)} \tag{5}$$

where ρ_n is the probability that the nth bit is zero. The maximum likelihood estimates for the parameters ρ_n are obtained simply from frequency counts down the columns $\mathbf{X}_l(:,n)$ for all l in the data set D, and dividing by the total number of rows in D.

We constructed this model based on the data and have plotted several key statistics in Figure ??. The plot indicates the negative log-likeihood of the data given the Bernoulli model 5. The negative log-likelihood is computed by taking the negative log of Equation 4.

This plot is similar to those found in the change-point detection or the process control literature, in that it shows the likelihood of observations as a function of time. Observations that fall outside of the "expected" bounds can be tagged for further observation. As noted before, this model completely ignores the potential relationship between the binary sensors or switches, as indicated in the product above.

DEPENDENT SENSORS INDEPENDENT OF TIME

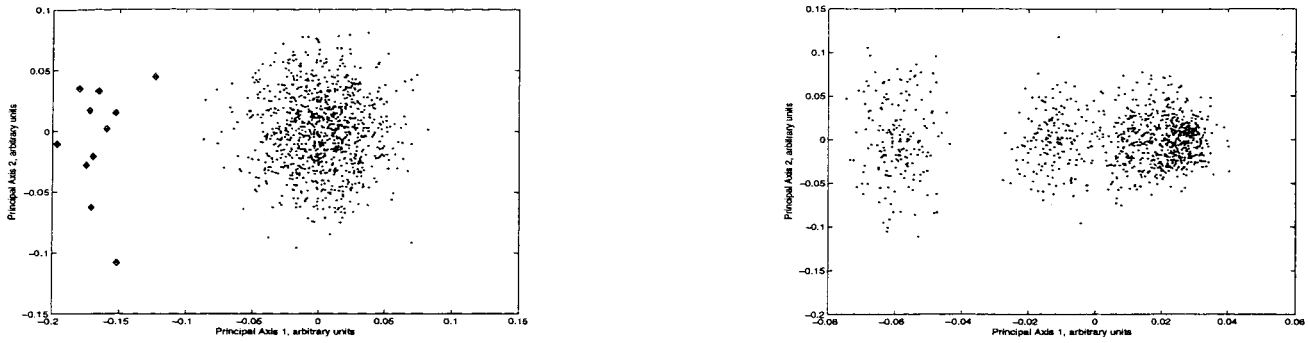

Figure 2: Using the same data sets as in Figure 1, we plot the two dimensional representation of the data points as projected onto the first two principal components. The data points corresponding to the 'highly unlikely' values above fall outside the modal region in this plot. These points are indicated by diamonds. The benchmark data set (left) clearly shows the failure modes. However, this transformation does not show any signs of failure for the dynamic data set.

Another approach that is often taken to analyze such data is based on the singular value decomposition (SVD) of the data set $\mathbf{X}_l$ [4]. This method works by factoring the matrix $\mathbf{X}_l$ into three matrices as follows:

$$\mathbf{X}_l = \mathbf{U}\Sigma\mathbf{V}^T \tag{6}$$

The matrix $\mathbf{U}$ is a $T \times N$ orthogonal matrix, Σ is an $N \times N$ diagonal matrix, and $\mathbf{V}$ is an $N \times T$ orthogonal matrix. SVD (which is closely related to principal components analysis) identifies the directions of maximum variation in the group of points defined by the rows of the data matrix. These directions can be shown to be the eigenvectors of the associated covariance matrix. Once the top m eigenvectors are identified (these correspond to those with the largest m eigenvalues), the observation vectors are left-multiplied with the eigenvectors. This results in an m dimensional representation of the observation vectors, where $m < N$. The parameter m is chosen in order to explain the maximum amount of variation in the data with the minimum number of eigenvectors.

It has been shown that plotting the first two columns of the unitary matrix $\mathbf{U}$ can give diagnostic information about the individual data points in the original data matrix [8]. We next demonstrate the effect of analyzing these binary streams using SVD. Figure 2 shows the distribution of points in $\mathbf{X}$ for the two benchmark and the dynamic data sets. The outlying points in left hand side of the figure correspond to the relatively unlikely events that occur near time step 500 in the likelihood time series.

These methods, while useful for analyzing high dimensional time series, do not capture the important dynamic aspects of the system. Instead, they treats each row in the time series matrix as an independent observation. More over, they amount to building an envelope detector in either the original space of the time series or a transformed space, as in the case of the SVD. We will next analyze a data set which is generated by a dynamical system which was a fault using dynamic models.

DYNAMIC MODELS

Of course one could choose to model all manner of dependencies between bits by building a full blown Bayesian network with both the structure of the dependency graph and the conditional probability tables are learned. However, since we must keep things computationally inexpensive we limit ourselves to tree models which can be learned in polynomial time. A comprehensive account of tree models can be found in Marina Meila's thesis available at `http://people.csail.mit.edu/people/mmp/thesis.html`. The simplest tree model could utilise the Chow-Liu algorithm [?]. That algorithm is straightforward and efficient. The

model has the form $P(\mathbf{x}) = \prod_{n=1}^{N} P(x(n)|x(\text{Pa}(n)))$ where $\text{Pa}(n)$ is the parent of variable $x(n)$.[3] The optimal parents can be determined by looking at the pairwise mutual informations with the details supplied in [?]. Miela gives improvements in the case where we have a very large number of columns of $\mathbf{X}$ (sensors).

0.1 Non-Stationary Aspects

In our approach regardless of how $P(\mathbf{x})$ is modelled time dependence is addressed by building a model for $P(\mathbf{X} = P(\mathbf{x}_1, \mathbf{x}_2, \cdots)$ for all rows (times). One reasonable modelling possibility here relies upon the Markov assumption where we assume that $\mathbf{x}_t$ is conditionally independent of all previous rows $\mathbf{x}_{t'}$ (with $t' < t$) except for x_{t-1}. In this case this simplification would give $P(\mathbf{X}) = P(\mathbf{x}_1) \prod_{t>1} P(\mathbf{x}_t|\mathbf{x}_{t-1})$. A naive hidden Markov model (HMM) is inappropriate to learn this model as any given row vector can assume 2^N possible values. Thus, we would first have reduce the number of features (perhaps by clustering with a mixture model) describing $P(\mathbf{x})$ and then build an HMM over these features.

Instead of pursuing this avenue we start with something simpler. We build a cruder description of the time dependence using a change point model. We assume that the model is stationary within different phases of the flight (perhaps takeoff, cruise, and landing), but that the behaviour between phases can differ. In the present context the change point model works as follows. For simplicity I assume 3 different phases T, C, and L but the idea is generalizable to any number of phases. In this case we also have very good ideas when the phases transition.

Assuming $t_2 > t_1$ divide the data so that $\tau_T = 1 : t_1$ is the takeoff phase T, $\tau_C = t_1 + 1 : t_2$ is the cruise phase C, and $\tau_L = t_2 + 1 : T$ is the landing phase.[4] Define $\mathbf{X}_f^T$, $\mathbf{X}_f^C$, $\mathbf{X}_f^L$ as $X(\tau_T, :)$, $X(\tau_C, :)$, and $X(\tau_L, :)$ respectively. Similarly $D^T = \{\mathbf{X}_1^T, \mathbf{X}_2^T, \cdots\}$, $D^C = \{\mathbf{X}_1^C, \mathbf{X}_2^C, \cdots\}$, and $D^L = \{\mathbf{X}_1^L, \mathbf{X}_2^L, \cdots\}$ are the corresponding data sets that can be used to train the models. Using these data sets we train a model exactly as using techniques described in Section ??. We call these models P^T, P^C, and P_L. The likelihood of any given $\mathbf{X} = [\mathbf{X}^T; \mathbf{X}^C; \mathbf{X}^L]$ is then

$$P(\mathbf{X}) = P^T(\mathbf{X}^T)P^C(\mathbf{X}^C)P^L(\mathbf{X}^L).$$

What remains then is how best to determine the phases, i.e. how to best determine the phase boundaries t_1 and t_2. We do this in the standard way by maximizing the log likelihood with respect to these unknown parameters. The data sets D^T, D^C, and D_L depend on t_1 and t_2 and so different choices will gives different values for the empirical log likelihood

$$E(t_1, t_2) \equiv \sum_f \log P(\mathbf{X}_f) = \sum_f \Big\{ \log P^T(\mathbf{X}^T) + \log P^C(\mathbf{X}^C) + \log P^L(\mathbf{X}^L) \Big\}.$$

Thus, we just optimize this quantity to determine the best boundaries. Naively, the optimization can be accomplished by trying a set of possible boundaries and selecting the best from amongst this set. Less naively, we might hillclimb to a local peak in log likelihood under a neighbourhood where the neighbours of (t_1, t_2) are $(t_1 + 1, t_2)$, $(t_1 - 1, t_2)$, $(t_1, t_2 + 1)$, $(t_1, t_2 - 1)$.[5] In this particular case where the phases correspond to takeoff, cruise, and landing we have a good idea where the appropriate boundaries might be and we can focus the optimization in this area.

References

[1] L. Atlas and et. al. Bloor, G., *An evolvable tri-reasoner ivhm system*, ISIS Vanderbilt Website (1999).

[2] A. Banerjee, I. Dhillon, J. Ghosh, and S. Sra, *Generative model-based clustering of directional data*, 2003.

[3] Note that some variables may not have parents in which case $\text{Pa}(n)$ is empty.

[4] I use the Matlab notation $a : b$ to indicate the set $[a, a + 1, \cdots, b - 1, b]$.

[5] Of course we would have to respect t_1 and t_2 being between 1 and T and having $t_2 > t_1$.

[3] K. R. Pattipati J. Ying, T. Kirubarajan and A. Patterson-Hine, *A hidden markov model-based algorithm for online fault diagnosis with partial and imperfect tests*, IEEE Transactions on SMC: Part C **30** (2000), no. 4, 463–473.

[4] I. T. Jolliffe, *Principal component analysis*, Springer, 2002.

[5] Eamonn J. Keogh, Selina Chu, David Hart, and Michael J. Pazzani, *An online algorithm for segmenting time series*, ICDM, 2001, pp. 289–296.

[6] M. Last, Y. Klein, and A. Kandel, *Knowledge discovery in time series databases*, 2001.

[7] L. R. Rabiner, *A tutorial on hidden markov models and selected applications in speech recognition*, Proceedings of the IEEE **77** (1989), no. 2, 257–286.

[8] D. B. Skillicorn, *Clusters within clusters: Svd and counterterrorism*, SIAM Workshop on Counterterrorism (2003).